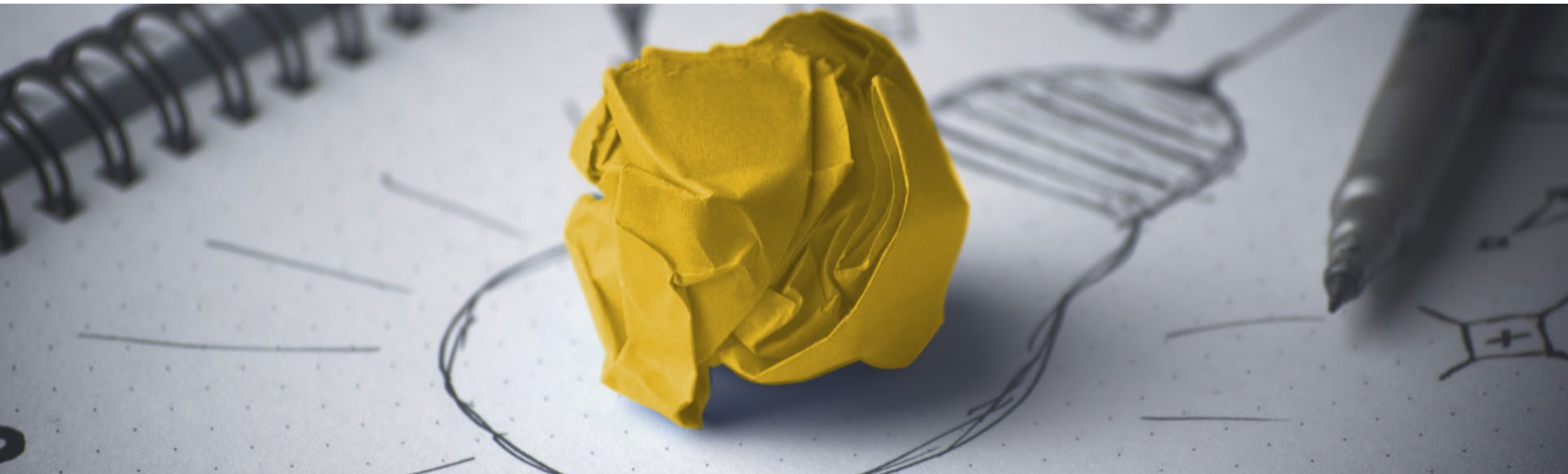


# Angular Beschleuniger

Angular erleben und anwenden



# Ziel von heute

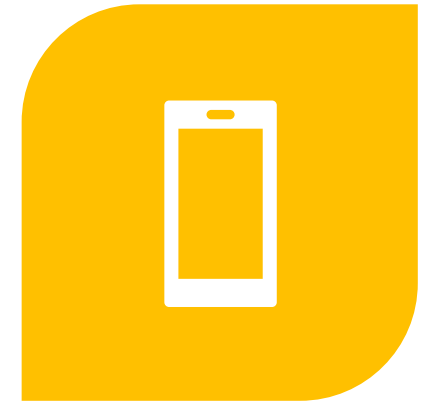
---



ANGULAR ERLEBEN /  
ANWENDEN



ANGULAR CLI ANWENDEN

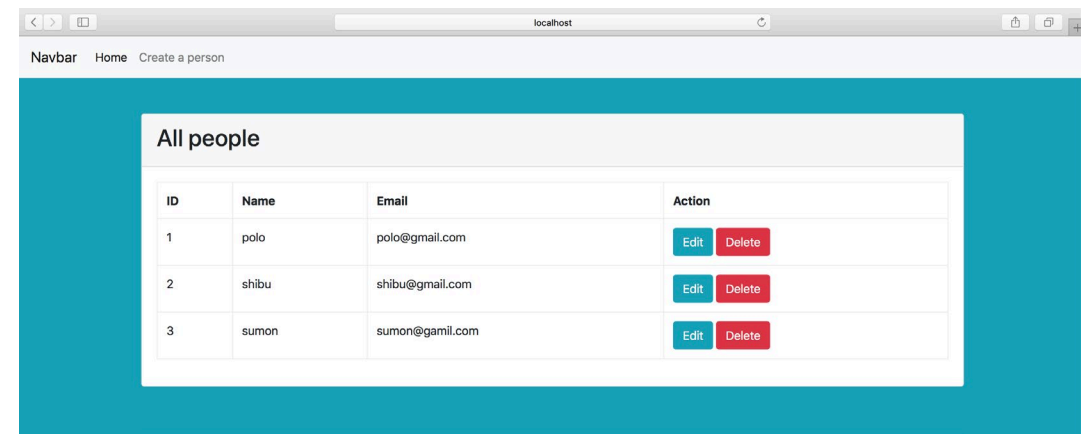


EIGENE MINI APPLIKATION  
GEBAUT HABEN

- Projekt Ziel
- Einrichtung Plugins Visual Studio Code
- Projekt start mit Angular CLI
  - Befehle für's starten
- Grundgerüst bauen
  - Debuggen (VS Code)
- Model Abbilden mit TypeScript
- Ausgeben des Modells im UI (Einfaches Binding – Property Binding)
- Werte via Formular verknüpfen (Two Way Binding - ngModel)
- Click Binding (Event Binding)
- Komponentisierung
- Navigation
- uvm.

# Wann setzen wir Angular ein?

- CRUD Anwendungen
- Fixe Framework Struktur
- Strukturierungsmöglichkeiten
  - Module (Modularisierung)
  - Komponenten (UI / Logik)
  - Services (Logik)



# Projekt Ziel

- CRUD Anwendung ;-)
- Pizza Bestellprozess
- Autom. kalk. des Preises
- Navigation
- Bestellen -> Bestellt die Pizza!
- Berechnen der Backzeit
- Dynamisches hinzufügen von Pizzas

- [Bestellung](#)
- [Kontakt](#)

Router Outlet:

## Bestellformular

Email:

- Margaritha (15 CHF) x
- Prosciutto (18 CHF) x

Total: **84**

# Lets GO!

---

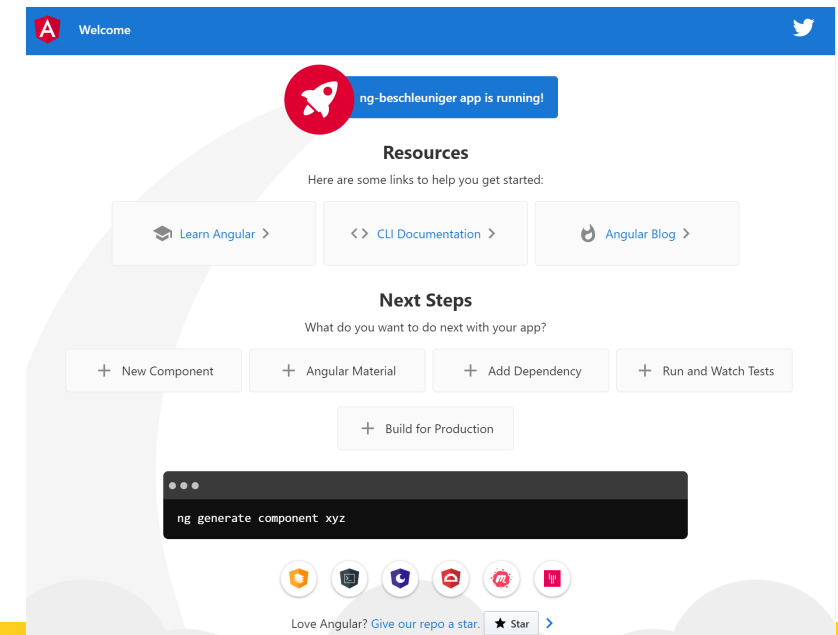
- Was brauchen wir
  - Angular CLI - <https://cli.angular.io/>
    - npm install -g @angular/cli
  - Visual Studio Code - <https://code.visualstudio.com/>
    - Plugins
      - Angular Essentials (Version 9) – John Papa
      - Angular Language Service
      - Angular Snippets (Version 9)
      - Debugger for Chrome
      - TSLint
      - «Angular Files»

# Build the app

- CMD / Terminal

```
D:\git>ng new ng-beschleuniger  
? Would you like to add Angular routing? (y/N) y
```

- SCSS auswählen
- Warten bis fertig!
- `cd ng-beschleuniger`
- `ng serve`
- <http://localhost:4200> voila



# Visual Studio Code - Shortlist

---

Show All Commands  +  + 

Go to File  + 

Find in Files  +  + 

Start Debugging 

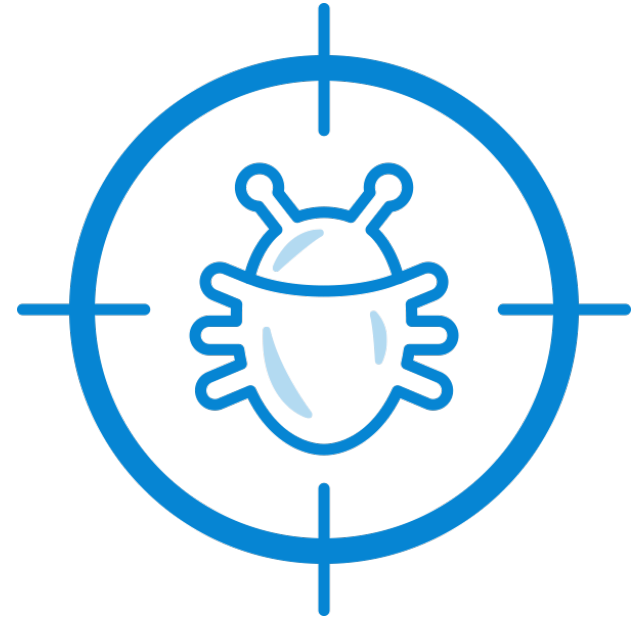
Toggle Terminal  + 



# Laufen lassen + Debug

---

- ng serve – Im Terminal
- ng t - Tests
- Und weitere - <https://angular.io/cli>
- Debug im VS Code einrichten
  - Breakpoint setzen
  - Launchconfig anpassen auf Port 4200



# Der Start – AppComponent - Titel

---

- index.html -> `<app-root></app-root>`
- Selector 'app-root' -> `app.component.ts`
- Allen bestehenden Code löschen & Titel mit «Bestellformular» anzeigen

# Modell – Pizza Liste

- Liste anzeigen

```
pizzas: Pizza[] = [  
    {name: 'Margaritha', price: 15},  
    {name: 'Prosciutto', price : 18},  
    {name: 'Hawaii', price: 20}  
];
```

## Bestellformular

```
<h1>Bestellformular</h1>
```

```
<ul>  
  <li *ngFor="let pizza of pizzas">{{ pizza.name }}</li>  
</ul>
```

- Margaritha
- Prosciutto
- Hawaii

# Implement Order Model

## Model

```
export class Pizza {  
  name: string;  
  price: number;  
}  
  
export class OrderItem {  
  amount: number;  
  pizza: Pizza;  
}  
  
export class Order {  
  orderItems: OrderItem[];  
  email: string;  
}
```

## Data Model

```
order: Order = {  
  email: '',  
  orderItems: [  
    { amount: 0, pizza: {name: 'Margaritha', price: 15}},  
    { amount: 0, pizza: {name: 'Prosciutto', price: 18}},  
    { amount: 0, pizza: {name: 'Hawaii', price: 20}},  
  ]  
};
```

# Order Model View

## Bestellformular

- Margaritha x 0
- Prosciutto x 0
- Hawaii x 0

## View

```
<ul>
  <li *ngFor="let orderItem of order.orderItems">{{ orderItem.pizza.name }} x {{ orderItem.amount }}</li>
</ul>
```

## Bestellformular

- Margaritha x 0 = 0
- Prosciutto x 1 = 18
- Hawaii x 2 = 40

```
<ul>
  <li *ngFor="let orderItem of order.orderItems">{{ orderItem.pizza.name }} x {{ orderItem.amount }} =
    {{ orderItem.pizza.price * orderItem.amount }}</li>
</ul>
```

# Order Item View

# Bestellformular

Email: meine@email.com

- Margaritha x 0 = 0
- Prosciutto x 1 = 18
- Hawaii x 2 = 40

```
order: Order = {
  email: 'meine@email.com',
  orderItems: [
    { amount: 0, pizza: {name: 'Margaritha', price: 15}},
    { amount: 1, pizza: {name: 'Prosciutto', price: 18}},
    { amount: 2, pizza: {name: 'Hawaii', price: 20}},
  ]
};
```

```
<h1>Bestellformular</h1>
```

```
<div>Email: {{ order.email }}</div>
```

```
<ul>
```

```
  <li *ngFor="let orderItem of order.
```

```
    {{ orderItem.pizza.price * orderI
```

```
</ul>
```

# Order Total – Extend Order Object (Getter)

```
export class Order {
  orderItems: OrderItem[];
  email: string;
```

Email: mymail@gmail.com

- Margaritha x 0 = 0
- Prosciutto x 1 = 18
- Hawaii x 2 = 40

Total: **58**

```
  get orderTotal(): number {
    let total = 0;
    this.orderItems.forEach(orderItem => {
      total += orderItem.amount * orderItem.pizza.price;
    });
    return total;
  }
}
```

```
order = new Order();
```

```
constructor() {
  this.order.email = 'mymail@gmail.com';
  this.order.orderItems = [
    { amount: 0, pizza: { name: 'Margaritha', price: 15 } },
    { amount: 1, pizza: { name: 'Prosciutto', price: 18 } },
    { amount: 2, pizza: { name: 'Hawaii', price: 20 } },
  ];
}
```

```
<div>Total: <strong>{{ order.orderTotal }}</strong></div>
```

# Email als Edit-Feld

## Bestellformular

Email:

- Margaritha x 0 = 0
- Prosciutto x 1 = 18
- Hawaii x 2 = 40

Total: **58**

```
<h1>Bestellformular</h1>
```

```
<div>Email: <input type="text" [value]="order.email" /></div>
```

```
<ul>
```

```
  <li *ngFor="let orderItem of order.orderItems">{{ orderItem.pizza.n
    {{ orderItem.pizza.price * orderItem.amount}}</li>
```

```
</ul>
```



# Anzahl Pizza anpassen

## Bestellformular

- Updates funktionieren nicht!
- NgModel ->

```
import { FormsModule } from '@angular/forms';
```

```
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
```

You, a few seconds ago | 1 author (You)

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

```
<h1>Bestellformular</h1>
```

```
<div>Email: <input type="text" [value]="order.email" /></div>
```

```
<ul>
```

```
  <li *ngFor="let orderItem of order.orderItems">{{ orderItem.pizza.name }} x
    <input type="number" [(ngModel)]="orderItem.amount" min="0"> =
    {{ orderItem.pizza.price * orderItem.amount}}</li>
```

```
</ul>
```

```
<div>Total: <strong>{{ order.orderTotal }}</strong></div>
```

Email:

- Margaritha x  = 0
- Prosciutto x  = 36
- Hawaii x  = 180

Total: **216**

# Bestellung absenden / Click Binding

- Daten an Server schicken
- Aktion auslösen

## Bestellformular

Email:

- Margaritha x  = 0
- Prosciutto x  = 18
- Hawaii x  = 40

Total: **58**

**Bestellen!**

`<button type="button" (click)="submitOrder()">Bestellen!</button>`

```
submitOrder() {  
  alert('Pizzas wurden bestellt an ' + this.order.email + '!');  
}
```

# Refactoring / Komponentisierung

- Auslagerung des Order Item
- Erstellung einer separaten Komponente für die OrderItem

```
<h1>Bestellformular</h1>
```

```
<div>Email: <input type="text" [(ngModel)]="order.email" /></div>
```

```
<ul>
```

```
  <li *ngFor="let orderItem of order.orderItems">{{ orderItem.pizza.name }} x  
    <input type="number" [(ngModel)]="orderItem.amount" min="0"> =  
    {{ orderItem.pizza.price * orderItem.amount}}</li>
```

```
</ul>
```

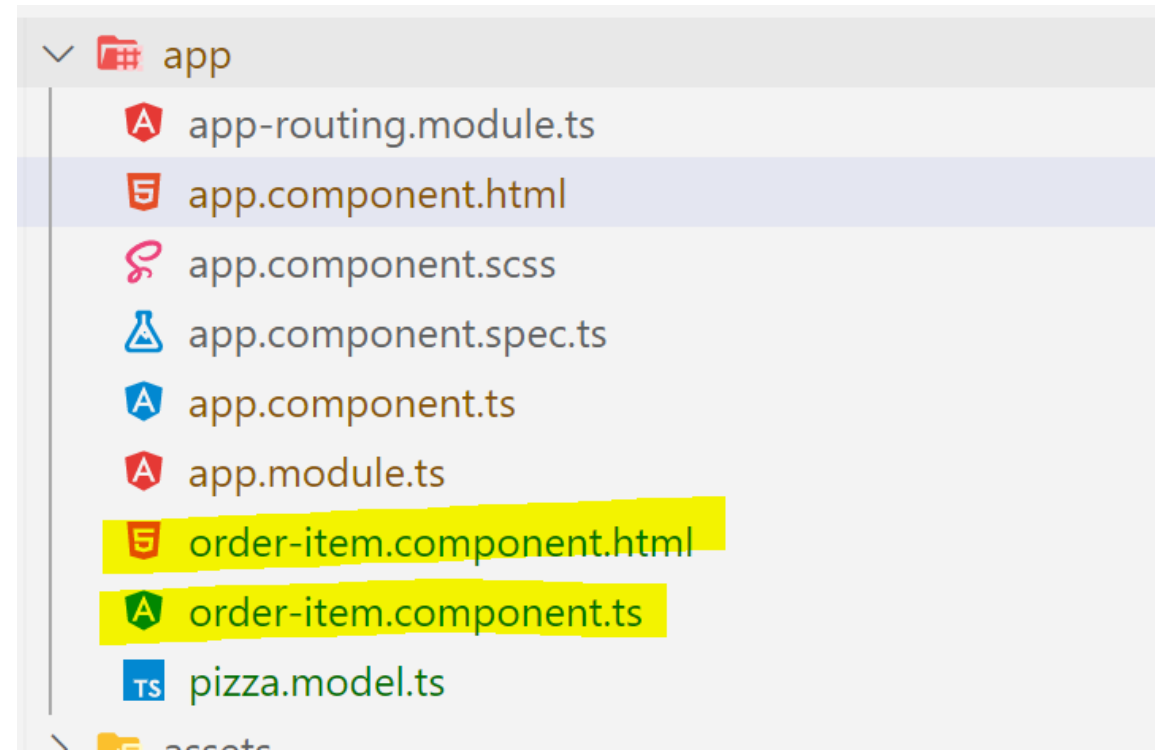
```
<div>Total: <strong>{{ order.orderTotal }}</strong></div>
```

```
<button type="button" (click)="submitOrder()">Bestellen!</button>
```

**order-item.component.ts**

# Neue Komponente erstellen

- Files erstellen



# TS File konfigurieren

## order-item.component.ts

- Minimal

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-order-item',
  templateUrl: 'order-item.component.html'
})

export class OrderItemComponent implements OnInit {
  constructor() { }

  ngOnInit() { }
}
```

**View**

# View konfigurieren

---

## order-item.component.html

```
app >  order-item.component.html > ...
```

```
<h1>TEST</h1>
```

```
|
```

# Komponente anzeigen

- Selector einfügen im app.component.html
  - `<app-order-item></app-order-item>`

```
<div>Email: <input type="text" [(ngModel)]="order.email" /></div>
<app-order-item></app-order-item>
<ul>
  <li *ngFor="let orderItem of order.orderItems">{{ orderItem.pizza.name }} x
    <input type="number" [(ngModel)]="orderItem.amount" min="0" > =
    {{ orderItem.pizza.price * orderItem.amount}}</li>
</ul>
```

- Komponente registrieren im Modul

```
import { OrderItemComponent } from './order-item.cc'
```

You, a few seconds ago | 1 author (You)

```
@NgModule({
  declarations: [
    AppComponent,
    OrderItemComponent
  ],
```

# Komponenten Kommunikation

```
<div>Email: <input type="text" [(ngModel)]="order.email" /></div>
```

```
<ul>
  <app-order-item *ngFor="let orderItem of order.orderItems" [orderItem]="orderItem"></app-order-item>
</ul>
```

```
@Component({
  selector: 'app-order-item',
  templateUrl: 'order-item.component.html'
})
```

```
export class OrderItemComponent implements OnInit {
  @Input() orderItem: OrderItem;
```

## order-item.component.html

```
<li>{{ orderItem.pizza.name }} x
  <input type="number" [(ngModel)]="orderItem.amount" min="0"> =
  {{ orderItem.pizza.price * orderItem.amount}}</li>
```



# Service

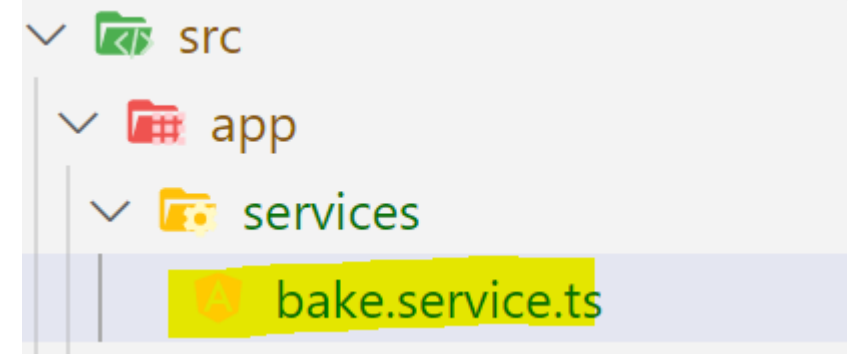
- Bake Timer
  - Anzahl bestellte Pizzas \* 5min
- A-Service

```
export class AppComponent {
  title = 'ng-beschleuniger';

  order = new Order();

  constructor(private bakeService: BakeService) {
    ...
  }

  submitOrder() {
    alert('Pizzas wurden bestellt an ' + this.order.email + '!');
    alert('Die Bestellung wird in ' +
      this.bakeService.calculateBakeTimeForOrder(this.order) +
      ' Minuten fertig gebachen sein');
  }
}
```



```
import { Injectable } from '@angular/core';
import { Order } from '../pizza.model';
```

```
@Injectable({ providedIn: 'root' })
export class BakeService {
  constructor() { }

  public calculateBakeTimeForOrder(pizzaOrder: Order) {
```

So ist die Registration im Modul nicht nötig

# Service im Modul registrieren

- Explizite Registration im Modul

```
@NgModule({  
  declarations: [  
    AppComponent,  
    OrderItemComponent  
  ],  
  imports: [  
    BrowserModule,  
    FormsModule,  
    AppRoutingModule  
  ],  
  providers: [BakeService],  
  bootstrap: [AppComponent]  
})
```

```
import { Injectable } from '@angular/core';  
import { Order } from '../pizza.model';
```

```
@Injectable()  
export class BakeService {  
  constructor() { }
```

# Output - Eventbinding

## app.component.html

```
<ul>
  <app-order-item *ngFor="let orderItem of order.orderItems"
    [orderItem]="orderItem"
    (orderAmountChanged)="orderAmountChanged($event)">
</ul>
```

## app.component.ts

```
orderAmountChanged(newOrderAmount: number) {
  console.log('new order amount: ' + newOrderAmount);
}
```

## order-item.component.html

```
<li>{{ orderItem.pizza.name }} x
  <input type="number" [(ngModel)]="orderItem.amount" min="0"
    (change)="amountChanged()"> =
  {{ orderItem.pizza.price * orderItem.amount}}</li>
```

## order-item.component.ts

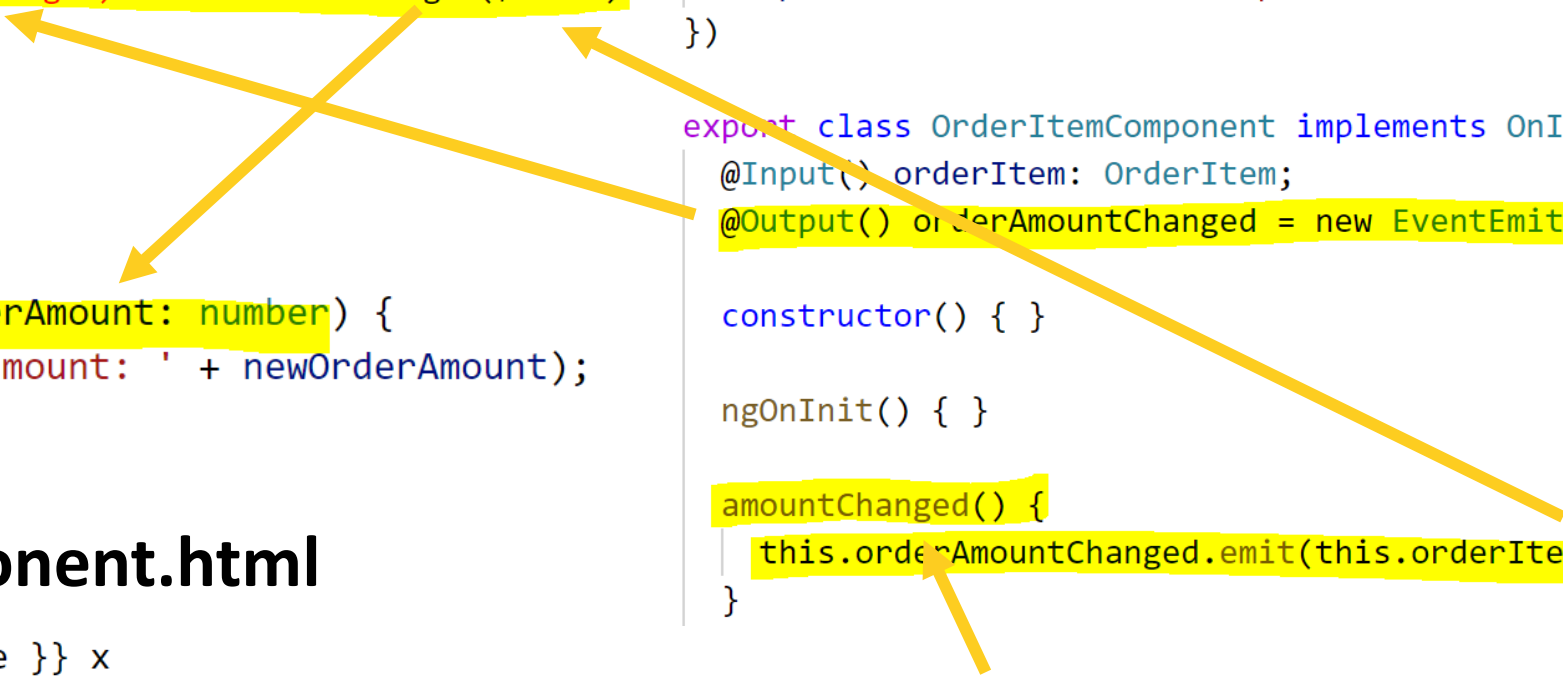
```
@Component({
  selector: 'app-order-item',
  templateUrl: 'order-item.component.html'
})

export class OrderItemComponent implements OnInit {
  @Input() orderItem: OrderItem;
  @Output() orderAmountChanged = new EventEmitter();

  constructor() { }

  ngOnInit() { }

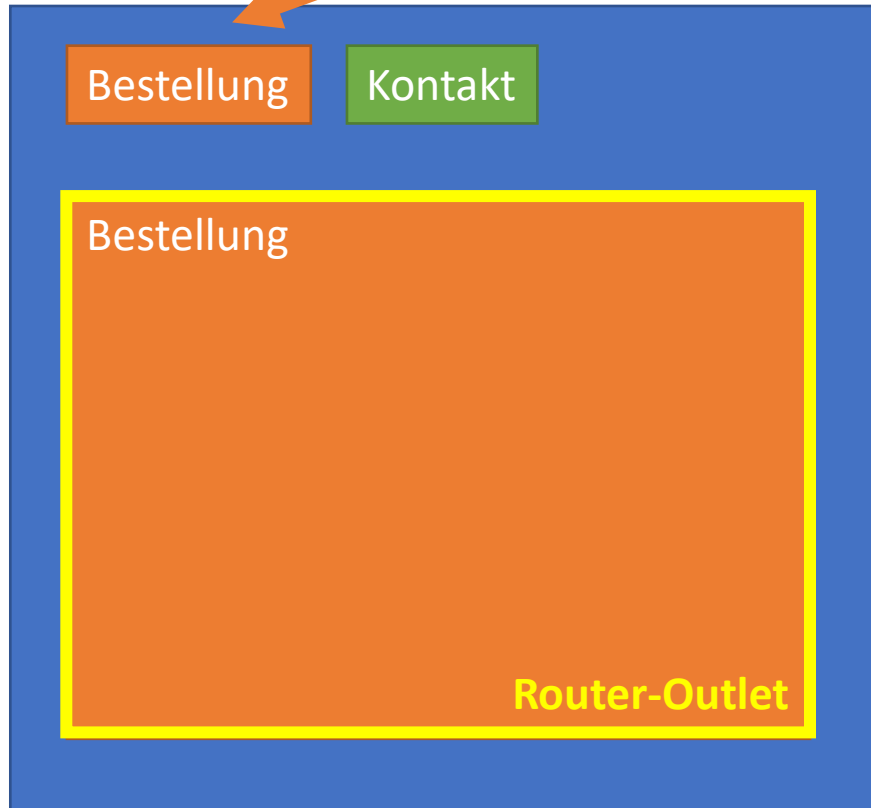
  amountChanged() {
    this.orderAmountChanged.emit(this.orderItem.amount);
  }
}
```



# Routing

```
const routes: Routes = [
  { path: 'order', component: OrderComponent },
  { path: 'contact', component: ContactComponent },
];
```

**app.component.html**



```
<ul>
  <li>
    <a [routerLink]="['/order']" >Bestellung</a>
  </li>
  <li>
    <a [routerLink]="['/contact']">Kontakt</a>
  </li>
</ul>
```

# Routing – Einfügen Router Outlet

- app.component.html

```
<button type="button" (click)="submitOrder()">Bestellen!</button>
```

```
<div>Router Outlet:</div>
<router-outlet></router-outlet>
```

## app-routing.module.ts

```
const routes: Routes = [];
```

You, 21 days ago | 1 author (You)

```
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

## app.module.ts

You, an hour ago | 1 author (You)

```
@NgModule({
  declarations: [
    AppComponent,
    OrderItemComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule
  ],
```

# Vorbereitung Ausbau Routing

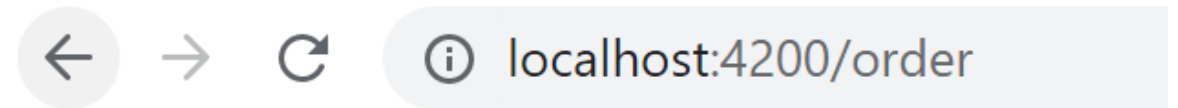
- Menüpunkte
  - Bestellung
  - Kontakt
- Neue Komponenten erstellen
  - order.component.ts
  - contact.component.ts

## app-routing.module.ts

```
const routes: Routes = [
  { path: 'order', component: OrderComponent },
  { path: 'contact', component: ContactComponent },
];
```

```
rc > app >  order.component.html > ...
```

```
1 <h1>Order</h1>
2 |
```



- [Bestellung](#)
- [Kontakt](#)

## Bestellformular

Email:

- Margaritha x  = 0
- Prosciutto x  = 18
- Hawaii x  = 40

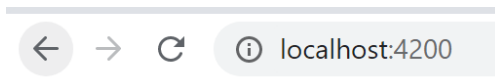
Total: **58**

Router Outlet:

**Order**

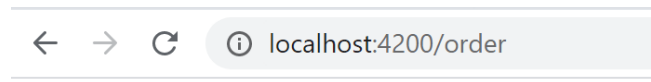
- Code und Markup

- app.component.ts -> order.component.ts
- app.component.html -> order.component.html (ausser `<router-outlet>`)



- [Bestellung](#)
- [Kontakt](#)

Router Outlet:



- [Bestellung](#)
- [Kontakt](#)

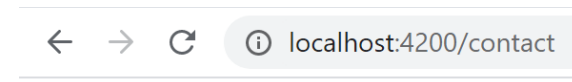
Router Outlet:

## Bestellformular

Email:

- Margaritha x  = 0
- Prosciutto x  = 18
- Hawaii x  = 40

Total: **58**



- [Bestellung](#)
- [Kontakt](#)

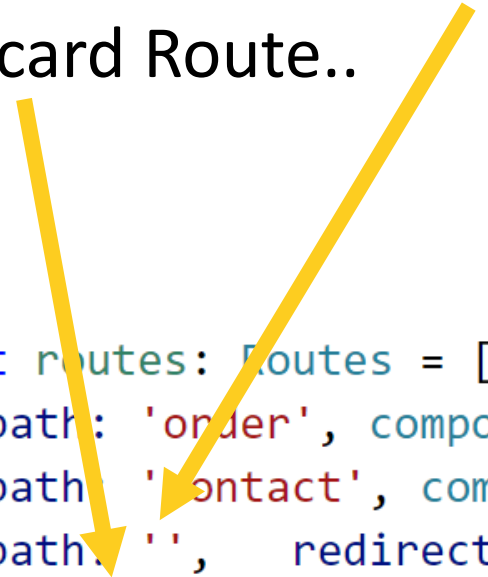
Router Outlet:

## Kontakt

# Einfügen der Default Route

- Leere Route weiterleiten
- Wildcard Route..

```
7 const routes: Routes = [  
3   { path: 'order', component: OrderComponent },  
3   { path: 'contact', component: ContactComponent },  
3   { path: '', redirectTo: '/order', pathMatch: 'full' },  
1   { path: '**', component: OrderComponent }, // 404 Component or Redirect  
2 ];
```





# Einzelne OrderItems «dynamisch hinzufügen»

- Bestehende Order Items entfernen (order.component.ts)

```
<button type="button" (click)="addOrderItem()">Order Item hinzufügen</button>
```

```
addOrderItem() {  
  this.order.orderItems.push({ amount: 0, pizza: { name: '', price: 0 } });  
}
```

```
<li>  
  <select [(ngModel)]="orderItem.pizza">  
    <option *ngFor="let pizza of availablePizzas" [ngValue]="pizza">{{ pizza.name }}</option>  
  </select>  
  
  {{ orderItem.pizza.name }} ({{ orderItem.pizza.price }} CHF) x  
  <input  
    type="number"  
    [(ngModel)]="orderItem.amount"  
    min="0"  
    (change)="amountChanged()"  
  />  
  = {{ orderItem.pizza.price * orderItem.amount }}  
</li>
```

```
export class OrderItemComponent implements OnInit {  
  @Input() orderItem: OrderItem;  
  @Output() orderAmountChanged = new EventEmitter();  
  
  availablePizzas: Pizza[] = [  
    { name: 'Margaritha', price: 15 },  
    { name: 'Prosciutto', price: 18 },  
    { name: 'Hawaii', price: 20 }];  
}
```

- [Bestellung](#)
- [Kontakt](#)

Router Outlet:

## Bestellformular

Email:

- ▼ Margaritha (15 CHF) x  = 30
- ▼ Prosciutto (18 CHF) x  = 54

Total: **84**

# Was haben wir gemacht?

---

- App from Scratch (Angular CLI)
- Entwicklungsumgebung Visual Studio Code & Plugins
- \*ngFor
- Komponenten erstellt
  - Kommunikation zwischen Komponenten (input / output binding)
- Service / Dependency Injection
- Navigation / Routing
- Dynamisches Formular

- <https://github.com/fxberry/pizza-order>

Es muss doch  
eine Lösung  
geben!

... oder?

